This article was originally published in the July, 2004 issue of FoxTalk 2.0
www.foxtalknewsletter.com

Introducing Inno Setup

Rick Borup

Inno Setup is a free, yet powerful, installer well suited for deploying Visual FoxPro applications. It provides a robust feature set for the developer and delivers a professional-looking installation for the end user. Rick Borup introduces you to Inno Setup and shows you how to use it with Visual FoxPro. This article covers the basic concepts and walks you through building a simple setup.

It's been said there's no such thing as a free lunch. That may be true, but there is such a thing as a free installer and its name is Inno Setup. The fact it's free is just icing on the cake, because Inno Setup is a great tool and would be well worth your time to learn even if you had to pay for it (which of course you can if you want to help support it).

Inno Setup is described as a script-based setup tool because the instructions you write to create a setup package are stored in a simple text, or script, file. Inno Setup scripts are easy to understand because they're based on a simple syntax that uses self-explanatory English words and abbreviations. You'll be able to create your first script in no time.

To get started with Inno Setup, download a copy from www.jrsoftware.org/isdl.php and install it on your computer. I'm using version 4.1.3 for this article, but Inno Setup is updated frequently and it's very likely there will be a newer version by the time you read this. Download and install the newest version available. The Inno Setup Web site has a complete revision history at www.jrsoftware.org/files/is4.2-whatsnew.htm where you can find out what's new in each release.

Although commonly referred to simply as Inno Setup, the formal name of this tool is the Inno Setup Compiler. The compiler, of course, is actually the most important part because it's what builds the setup package. After installing Inno Setup, you have an *Inno Setup 4* group on your Start menu. To launch Inno Setup, choose *Inno Setup Compiler* from this menu. In addition to the compiler, Inno Setup has its own editor for creating and editing script files.

Inno Setup comes with a good Help file. Unfortunately it's in the older WinHelp format instead of HTML Help, but the quality of the content makes up for this minor inconvenience. After you install Inno Setup, take a moment to familiarize yourself with the Help file. For starters I'd recommend reading the "What is Inno Setup?" page and the first four pages under the "How to Use" section. Then locate the "Setup Script Sections" section of the Help file and keep it handy as you work and learn Inno Setup. This section is your primary reference for Inno Setup script syntax, and you may want to refer to it as you read the rest of this article.

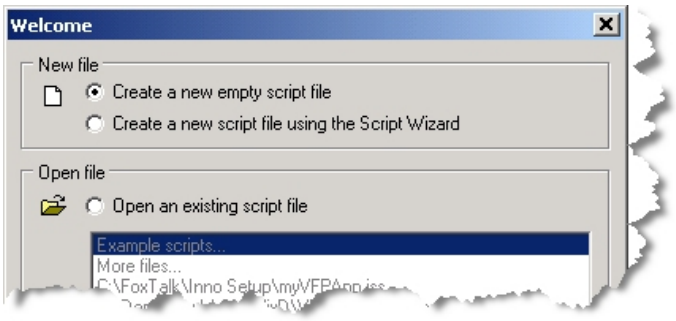
Creating your first Inno Setup script

This article takes you through the steps necessary to create a minimal yet fully functional Inno Setup script for a hypothetical Visual FoxPro 8.0 application called myVFPApp. The distributables for myVFPApp are an executable file named myVFPApp.exe and a readme file named readme.txt. In addition, the setup will need to install the Visual FoxPro runtime files and their dependencies.

Inno Setup script files are text files with an ISS file name extension. Because they are simply text files, you can create or edit Inno Setup scripts with any text editor. Using Inno Setup's own built-in editor has several advantages, though, among them syntax coloring and one-button access to the compiler.

When you launch Inno Setup, a Welcome dialog offers to create a new script or open an existing one, as illustrated in **Figure 1**. Choosing *Create a new empty script file* creates a brand new file that starts out as a blank page in the editor.

Figure 1



The other choice is to use the Script Wizard, a built-in tool that walks you through a series of steps to create a new setup script. You should be aware of the Script Wizard because if it is turned on (as it is by default) it starts automatically when you choose File | New from the main menu. I won't cover the Script Wizard here. If you'd like to know more, you can explore it on your own; you can also find a discussion of the Script Wizard in Appendix D of *Deploying Visual FoxPro Solutions* from Hentzenwerke Publishing.

In addition to the usual buttons for New, Open, Save and Help, the toolbar of the InnoSetup IDE has buttons to start and stop the compiler and to run the setup after compilation.

An Inno Setup script file is organized into sections, much as an INI file. Section headers are enclosed in square brackets, and the entries for each section follow the section header with one entry per line. This simple structure makes it easy to read and write a script file.

The Setup section

The first section in an Inno Setup script is the Setup section. The entries in this section pertain to the entire setup and include settings for the name of the product, its version number, and so forth. The Setup section entries for myVFPApp are as follows. Note that you can place a comment anywhere in a script by starting the line with a semi-colon.

```
; -- myVFPApp.iss --
; This is the setup script for myVFPApp version 1.0.0.
[Setup]
AppName=MyVFPApp
AppVerName=MyVFPApp version 1.0.0
DefaultDirName={pf}\MyVFPApp
DefaultGroupName=MyVFPApp
```

The names to the left of the equals sign in the Setup section are called directives. As you can see, directives are easy-to-read English words or abbreviations. Although a typical script has many directives in the Setup section, only AppName, AppVerName, and DefaultDir are required. The AppName directive is the name of the application the way you want it to be displayed during installation. The AppVerName directive is similar to AppName, but includes the version number. The DefaultDirName directive specifies the default destination directory for the product. The other directive shown here is DefaultGroupName, which specifies the default Start menu folder name used when creating shortcuts.

In the DefaultDirName directive, notice the use of the constant {pf} in front of the directory name. Inno Setup uses several constants like this to indicate entries whose actual values are resolved at installation time. The {pf} constant points to the Program Files folder on the user's computer. This and the other constants used by Inno Setup are explained in the Help file under How to Use | Constants.

The Files section

The Files section is where you list the files you want to install. The entries in this section are structured in a *parameter: "value"* format. You can place two or more parameters on one line by separating them with semi-colons. The Files section entries for myVFPApp are as follows.

```
[Files]
Source: "MyVFPApp.exe"; DestDir: "{app}"
Source: "Readme.txt"; DestDir: "{app}"; Flags: isreadme
```

Entries in the Files section require a minimum of two parameters: a Source parameter and a DestDir parameter. The Source parameter specifies the location of the source file on your computer or network. Unless a full drive and path are specified, the source is relative to the location of the script file. The DestDir parameter identifies the destination directory for the file on the user's computer. Note the use of the constant {app} as the value for the DestDir parameter. The {app} constant points to the directory the user chooses as the destination directory when installing the application.

The entry for the readme.txt file illustrates the use of flags. Flags allow you to specify additional options for a file. The *isreadme* flag tells the compiler this is the "read me" file for this setup and instructs the installer to prompt the user to view the file after installation is complete. This flag can be used on only one file per script.

The Icons section

Except for the VFP runtime files, you could actually stop right here and have a complete script. It's customary to create a shortcut, though, and in Inno Setup this is accomplished by adding an Icons section. Entries in the Icons section, like the entries in the Files section, consist of parameter and value pairs. Following is the entry to create a Start Menu icon for myVFPApp.

```
[Icons]
Name: "{group}\MyVFPApp"; Filename: "{app}\MyVFPApp.exe"
```

Again, note the use of constants. The Name parameter uses the {group} constant, which points to the Start Menu folder on the user's computer. The Filename parameter uses the {app} constant along with the name of the application's executable file to specify the target of the shortcut.

The VFP runtime files

The script entries for the VFP runtime files belong in the Files section, but these files require special handling: the runtime files for VFP 7 and VFP 8 go into the Common Files folder, and the Visual C++ 7.0 runtime DLL goes into the System folder. Because the entries for the runtime files are the same for all VFP apps, you can save them in a separate file and simply copy them into your setup scripts as needed.

Here are the Files section entries for the VFP 8 runtime files. (Line breaks have been inserted to fit the column width for publication; in the actual script each entry is a single line.)

```
[Files]
Source: C:\Program Files\Common Files\Microsoft Shared\
VFP\gdiplus.dll; DestDir: {cf}\Microsoft Shared\VFP;
Flags: sharedfile uninsneveruninstall restartreplace

Source: C:\Program Files\Common Files\Microsoft Shared\
VFP\vfp8r.dll; DestDir: {cf}\Microsoft Shared\VFP;
Flags: regserver sharedfile uninsneveruninstall
restartreplace

Source: C:\Program Files\Common Files\Microsoft Shared\
VFP\vfp8t.dll; DestDir: {cf}\Microsoft Shared\VFP;
Flags: sharedfile uninsneveruninstall restartreplace

Source: C:\Program Files\Common Files\Microsoft Shared\
VFP\vfp8renu.dll; DestDir: {cf}\Microsoft Shared\VFP;
Flags: sharedfile uninsneveruninstall restartreplace

Source: C:\VFP8Distrib\System32\msvcr70.dll; DestDir: {sys};
Flags: uninsneveruninstall onlyifdoesntexist
```

Note the use of a fully qualified drive and path for the Source parameter. This is because the source files are not in the same directory as the script. The source for the VFP runtime files is the Common Files folder on your computer, which can be referenced directly in the script as shown. The msvcr70.dll file resides in your computer's System folder, but for safety reasons Inno Setup (by default) no longer allows you to deploy files directly from your own System folder. The solution in this case is to copy msvcr70.dll to another folder and reference the copy in your script.

Another new constant is introduced in the DestDir parameter for these entries. The {cf} constant points to the Common Files folder on the user's computer. New flags are introduced here, too. The *sharedfile* flag tells the installer to increment the file's reference count on the user's computer. The *uninsneveruninstall* flag tells the installer not to uninstall this file when the application is uninstalled. The *restartreplace* flag tells the installer to set the file to be replaced upon restart if the file is in use during installation. The *onlyifdoesntexist* flag means copy this file only if it doesn't already exist on the user's computer. Finally, the *regserver* flag (used with vfp8r.dll) tells the installer to register the DLL on the user's computer.

You can add these five entries to the Files section of your script by simply typing them in, or you can store them in a separate file and copy and paste them from there. A third way is to use Inno Setup's *#include* directive to bring them in from a separate file at compile time. For example, if you store these entries in a file called VFP8Runtimes.txt, you can include them in your script by adding the following line to the Files section:

```
#include "VFP8Runtimes.txt"
```

Files you incorporate into your script with the *#include* directive don't need to have an ISS file name extension, although they can. It's okay to have more than one section with the same name in a script, so you don't have to worry about removing the section header(s) from an included file.

Compiling the script

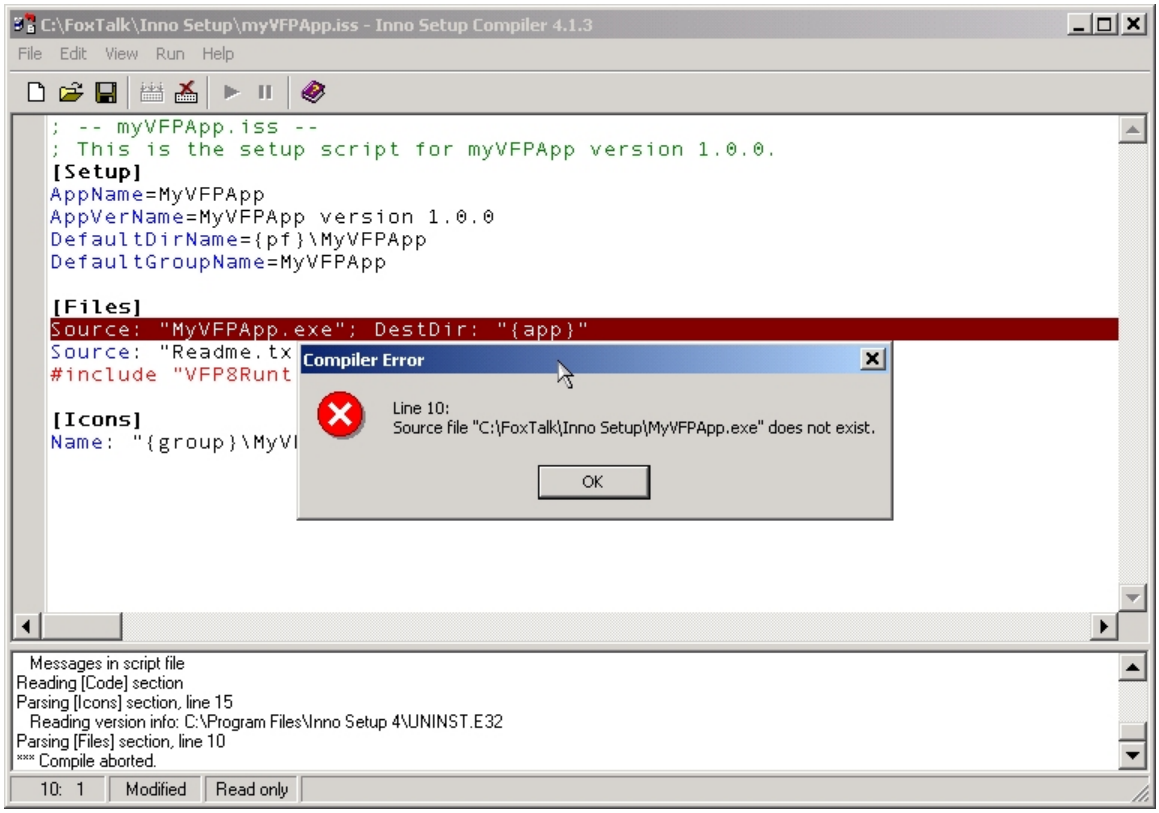
When you're ready to compile your script, simply click the Compile button on the Inno Setup toolbar as illustrated in **Figure 2**. Before compiling a new script for the first time, give it a name and save it to disk.

Figure 2

The sample script myVFPApp.ISS is available in accompanying download file. The Download also includes the VFP8Runtimes.txt file as well as myVFPApp.exe and readme.txt. To compile the script you need VFP 8 installed on your machine so the compiler can find the runtime files.

When you click the Compile button, Inno Setup compiles the script file line by line, opening a Compiler Status window below the editor to log each step in the compilation and any errors encountered. If an error is detected the compilation halts, an error message displays, and the line with the error will be highlighted in the editor (see **Figure 3**).

Figure 3



If no errors are detected, the compiler generates the setup.exe file. The usual location for this file is a folder called Output, which by default is created under the directory where the script itself is located. To deploy your app to your users, simply send them the setup.exe file.

Inno Setup ships with a number of sample scripts to help get you started. Although not specific to VFP, these sample scripts are a good place to start learning. If you install Inno Setup to its default location, the sample scripts are located in C:\Program Files\Inno Setup 4\Examples.

Stay tuned

In this article, I showed you how to create a simple Inno Setup script file for a Visual FoxPro application. In a future article, I'll show you some cool things you can do with your scripts to make working with them easier for you and to enhance the installation experience for your end users.

Download the code from www.ita-software.com/papers/FT407Borup.zip.

Rick Borup is an independent developer who has been working with FoxPro and Visual FoxPro for over ten years. He is owner and president of Information Technology Associates, a software design, development, and consulting firm he founded in 1993. Rick has spoken about VFP and related topics to several user group meetings and was a speaker at the Great Lakes Great Database Workshop in Milwaukee. He is an MCSD and MCP in Visual FoxPro, and is co-author of the book Deploying Visual FoxPro Solutions from Hentzenwerke Publishing. rborup@ita-software.com.